**1. What is React?**

**Answer**: React is a declarative, efficient, and flexible JavaScript library used for building user interfaces, particularly for single-page applications where you need a fast, interactive user experience. React allows the creation of reusable components and uses a virtual DOM for efficient updates.

**2. What are the key features of React?**

**Answer**: Key features of React include:

* **JSX**: A syntax extension for JavaScript that allows writing HTML-like code in JavaScript.
* **Components**: Reusable, independent units of code that define UI.
* **Virtual DOM**: A lightweight copy of the real DOM used to optimize UI updates.
* **Unidirectional Data Flow**: Data flows in one direction from parent to child components via props.
* **React Hooks**: Functions like useState, useEffect, etc., that enable state and lifecycle management in functional components.

**3. What is JSX in React?**

**Answer**: JSX is a syntax extension for JavaScript that looks like HTML but allows you to write components and UI elements in the same file as JavaScript code. JSX is compiled by Babel into React.createElement calls.

Example:

jsx

Copy code

const element = <h1>Hello, world!</h1>;

**4. What is a component in React?**

**Answer**: A component is a building block of a React application. A component can either be a **class component** or a **functional component**, and it manages the rendering of the UI based on input (via props and state).

* **Functional Component**: Simpler, written as functions, can use hooks to manage state.
* **Class Component**: Has lifecycle methods and internal state (pre-hooks).

**5. What is the difference between a class component and a functional component?**

**Answer**:

* **Class Component**: A class-based React component that extends React.Component. It has lifecycle methods like componentDidMount, componentWillUnmount, etc.

jsx

Copy code

class MyComponent extends React.Component {

render() {

return <h1>{this.props.name}</h1>;

}

}

* **Functional Component**: A simpler, function-based component that can use hooks for managing state and side effects.

jsx

Copy code

function MyComponent(props) {

return <h1>{props.name}</h1>;

}

**6. What are props in React?**

**Answer**: Props (short for properties) are read-only data passed from a parent component to a child component. Props are immutable and cannot be changed by the child component. They are used to configure or pass dynamic content.

jsx

Copy code

function Greeting(props) {

return <h1>Hello, {props.name}</h1>;

}

**7. What is state in React?**

**Answer**: State is an object that holds data that can change over time. State is local to the component, and when state changes, the component re-renders. State is mutable and can be updated using setState in class components or hooks like useState in functional components.

jsx

Copy code

const [count, setCount] = useState(0);

**8. What are React Hooks?**

**Answer**: React Hooks are functions introduced in React 16.8 that enable functional components to manage state and side effects. The most common hooks are:

* **useState**: To manage state in functional components.
* **useEffect**: To perform side effects such as data fetching or DOM manipulation.
* **useContext**: For accessing values from context without prop drilling.
* **useReducer**: For managing more complex state logic.

**9. What is useState in React?**

**Answer**: useState is a hook that allows functional components to have state. It returns an array containing the current state value and a function to update that value.

jsx

Copy code

const [count, setCount] = useState(0); // Initializes state with value 0

**10. What is useEffect in React?**

**Answer**: useEffect is a hook that handles side effects in functional components, such as fetching data, subscribing to events, or manually changing the DOM. It can be used for lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount in class components.

jsx

Copy code

useEffect(() => {

// Code to fetch data

console.log('Component mounted');

}, []); // Empty dependency array ensures it runs only once (on mount)

**11. What is the Virtual DOM in React?**

**Answer**: The Virtual DOM is a lightweight in-memory representation of the actual DOM. React updates the Virtual DOM first when the state of a component changes. React then compares the Virtual DOM with the previous state and updates the real DOM efficiently, minimizing the number of DOM manipulations.

**12. What is React Context?**

**Answer**: React Context provides a way to share values between components without having to explicitly pass props through every level of the component tree. It is often used for global data like authentication, themes, or language preferences.

jsx

Copy code

const ThemeContext = React.createContext('light');

**13. What is Redux?**

**Answer**: Redux is a state management library used to manage the state of large-scale applications. It helps in managing state in a predictable manner using a single global store, actions to describe changes, and reducers to apply state changes based on actions.

**14. What is the connect() function in React Redux?**

**Answer**: connect() is a higher-order component used in React-Redux to connect React components to the Redux store. It allows components to access store data via mapStateToProps and dispatch actions via mapDispatchToProps.

jsx

Copy code

const mapStateToProps = (state) => ({

count: state.count

});

const mapDispatchToProps = (dispatch) => ({

increment: () => dispatch({ type: 'INCREMENT' })

});

export default connect(mapStateToProps, mapDispatchToProps)(MyComponent);

**15. What are controlled components?**

**Answer**: A controlled component is an element (like <input>, <textarea>) whose value is controlled by React state. The value is stored in the component’s state, and React is responsible for updating the UI.

jsx

Copy code

const [value, setValue] = useState('');

return <input value={value} onChange={(e) => setValue(e.target.value)} />;

**16. What are uncontrolled components?**

**Answer**: An uncontrolled component is an element where the form data is handled by the DOM itself, not by React. Refs are used to interact with the DOM and get the value of the input.

jsx

Copy code

const inputRef = useRef(null);

<input ref={inputRef} />;

**17. What is Prop Drilling in React?**

**Answer**: Prop drilling refers to the process of passing data from a parent component down to deeply nested child components through intermediate levels of components. This can make the code harder to maintain.

**Solution**: Use **React Context** or **Redux** to avoid prop drilling by passing data globally to all components.

**18. What are keys in React?**

**Answer**: Keys are unique identifiers used by React to identify which items have changed, been added, or been removed from a list. They help in optimizing rendering by ensuring efficient updates to the UI.

jsx

Copy code

const listItems = items.map((item, index) => <li key={index}>{item}</li>);

**19. What are lifecycle methods in React?**

**Answer**: Lifecycle methods are methods in class components that allow you to run code at specific points in a component's life (e.g., mounting, updating, unmounting). Common lifecycle methods include:

* componentDidMount
* componentDidUpdate
* componentWillUnmount

With functional components, the useEffect hook is used to manage side effects.

**20. What is React Router?**

**Answer**: React Router is a library used for routing in React applications. It allows you to navigate between different components/views based on the URL.

jsx

Copy code

import { BrowserRouter as Router, Route } from 'react-router-dom';

<Router>

<Route path="/home" component={Home} />

<Route path="/about" component={About} />

</Router>

**21. What is shouldComponentUpdate in React?**

**Answer**: shouldComponentUpdate is a lifecycle method in class components that allows you to control whether a component should re-render when the state or props change. It helps optimize performance by preventing unnecessary re-renders.

**22. What is React.memo()?**

**Answer**: React.memo() is a higher-order component that memoizes the result of a functional component. It prevents unnecessary re-renders by only re-rendering the component if the props have changed.

jsx

Copy code

const MyComponent = React.memo(function MyComponent(props) {

// Component code here

});

**23. What is useCallback in React?**

**Answer**: useCallback is a hook that returns a memoized version of a callback function, which only changes if one of its dependencies has changed. It helps to avoid unnecessary re-creations of functions.

jsx

Copy code

const memoizedCallback = useCallback(() => {

// Code here

}, [dependency]);

**24. What is useMemo in React?**

**Answer**: useMemo is a hook that memoizes the result of a computation. It only recomputes the value when one of its dependencies changes, optimizing performance for expensive calculations.

jsx

Copy code

const memoizedValue = useMemo(() => expensiveFunction(count), [count]);

**25. What are fragments in React?**

**Answer**: Fragments are a way to group a list of children without adding extra nodes to the DOM. They are useful when returning multiple elements from a component but don't want to wrap them in a <div>.

jsx

Copy code

return (

<>

<h1>Title</h1>

<p>Some content</p>

</>

);

**26. What is the difference between componentDidMount and useEffect?**

**Answer**: componentDidMount is a lifecycle method in class components that is called once after the component is mounted. useEffect can be used in functional components to perform side effects, and it runs after every render, but can be configured to run only once or when specific dependencies change.

**27. What is the dangerouslySetInnerHTML attribute in React?**

**Answer**: dangerouslySetInnerHTML is a React attribute that allows you to set HTML content directly in a component. It’s called "dangerous" because it can lead to XSS (Cross-Site Scripting) vulnerabilities if you inject unsanitized user input.

jsx

Copy code

<div dangerouslySetInnerHTML={{ \_\_html: 'Hello <b>World</b>' }} />

**28. What is Server-Side Rendering (SSR) in React?**

**Answer**: SSR is the process of rendering React components on the server and sending the fully rendered page to the client, which improves performance and SEO. Tools like **Next.js** make it easy to implement SSR in React.

**29. What is Client-Side Rendering (CSR) in React?**

**Answer**: CSR is the process where the browser fetches the JavaScript and renders the React components on the client side. React is first loaded, and only then is the UI generated. CSR is typical in single-page applications (SPAs).

**30. What is the purpose of useRef in React?**

**Answer**: useRef is a hook used to persist values across renders without triggering re-renders. It can be used to access DOM elements or store mutable values.

jsx

Copy code

const inputRef = useRef();

**31. How do you handle forms in React?**

**Answer**: Forms in React are typically controlled components where form data is stored in the state, and changes are handled using event handlers like onChange.

jsx

Copy code

const [value, setValue] = useState('');

function handleChange(event) {

setValue(event.target.value);

}

return <input value={value} onChange={handleChange} />;

**32. What is a Higher-Order Component (HOC)?**

**Answer**: An HOC is a function that takes a component and returns a new component with additional props or functionality. HOCs are used for cross-cutting concerns like authentication, logging, etc.

jsx

Copy code

function withLogging(Component) {

return function WrappedComponent(props) {

console.log('Rendering component', Component.name);

return <Component {...props} />;

};

}